Design and FPGA Implementation of Stochastic Turbo Decoder

Quang Trung Dong[1], Matthieu Arzel[1] and Christophe Jégo[2]

[1]Institut Telecom, Telecom Bretagne, CNRS Lab-STICC UMR 3192
Technopôle Brest Iroise, CS 83818 29238 Brest
Université Européenne de Bretagne, France
firstname.lastname@telecom-bretagne.eu

[2]IPB, ENSEIRB-MATMECA, CNRS IMS, UMR 5218
351 Cours de la Libération, 33405 Talence
Université de Bordeaux, France
christophe.jego@ims-bordeaux.fr

ABSTRACT

Stochastic decoding that is inspired by stochastic computation is an alternative technique for decoding of error-correcting codes. The extension of this approach to decode convolutional codes and turbo codes is discussed in this article. The switching activity sensitivity is circumvented and the latching problem is reduced by transforming the stochastic additions into stochastic multiplications in the exponential domain and using multiple streams with deterministic shufflers. The number of decoding cycles is thus considerably reduced with no performance degradation. Stochastic decoding, previously applied to the decoding of LDPC codes, can now be applied to decoding of turbo codes. In addition, the first hardware architecture for stochastic decoding of turbo codes is presented. The proposed architecture makes fully-parallel turbo decoding viable on FPGA devices. Results demonstrate the potential of stochastic decoding to implement fully-parallel turbo decoders.

1. INTRODUCTION

Principles of stochastic computation were elaborated in the 1960's by Gaines [1] and Poppelbaum *et al.* [2] as a method to carry out complex operations with a low hardware complexity. The main feature of this method is that the probabilities are converted into streams of stochastic bits using Bernoulli sequences in which the information is given by the statistics of the bit streams. Complex arithmetic operations on probabilities such as multiplication and division are transformed into operations on bits using elementary logic gates. Stochastic decoding of Forward Error Correction (FEC) is inspired by these principles. The two main appealing features of this approach for iterative decoding are very simple hardware structures of computing nodes and high-throughput decoding. Early stochastic methods were only successful for decoding special short/acyclic Hamming [3] or LDPC codes [4]. These methods result in poor decoding performance when used to decode practical codes on factor graphs (with cycles). An improved stochastic decoding approach was then proposed to decode practical LDPC codes [5]. When compared with conventional Sum-Product implementations, stochastic decoding could provide near-optimal performance for practical LDPC codes. The potential of this method for low-complexity decoding was validated by an FPGA implementation of a (1024, 512) LDPC decoder. This first implementation provides a throughput of 706 Mbps at a Bit Error Rate (BER) of 10^{-6} with performance loss of about 0.1 dB, compared to the floating-point BP, while occupying only 36% of a Virtex-4 LX200 FPGA device [6]. High data rates and architecture efficiencies were achieved thanks to many recent improvements [7] [8]. In addition, this approach was extended to well-known linear block codes with high-density parity-check matrices, namely BCH codes, Reed Solomon codes and product codes [9]. Moreover, stochastic decoding is also of high interest for low power decoders [10] and fault-tolerant nanoscale circuits [11].

Turbo codes [12] are a family of FECs that are especially attractive for mobile communication systems and have been adopted as part of several channel coding standards for high data rates such as UMTS and CDMA2000 (third-generation) or 3GPP-LTE (the last step toward the fourth generation). A major challenge in the implementation of turbo decoders is high-throughput decoding. Indeed, the next generations of mobile communication systems will require data rates of 1 Gb/s and beyond. Designers try to exploit the maximum feasible amount of parallelism in turbo decoders for the sake of higher throughput. However, due to the lack of parallelism in the MAP-based decoding algorithm, the implementation of fully-parallel turbo decoders is still challenging for practical turbo codes.

Another form of stochastic algorithm was proposed in [13] and used for trellis decoding algorithm of an acyclic (16,11) Hamming code and a (256,121) product Turbo code based on 32 component decoders of this Hamming code. An improved stochastic decoding approach was recently introduced to decode convolutional codes and turbo codes [14]. The results provided in this paper validate the potential of stochastic decoding as a practical approach for high-throughput turbo decoders and encourage to keep on investigating in this way. One major problem in stochastic decoding which deeply degrades the performance is related to the sensitivity to the level of random switching activity. Different solutions such as Edge Memories (EMs) [7] have been suggested to solve this latching problem. In [15], authors introduced an original idea to reduce the number of clock cycles required to decode one codeword and to replace the EMs by simple deterministic shufflers. The proposed architecture, that uses multiple streams in parallel to represent the same probability, presents the best architecture

efficiency defined as the ratio between data throughput and hardware complexity.

Thanks to these previous studies, this paper discusses the implementation issues of stochastic turbo decoders. The first FPGA-based implementation of a stochastic turbo decoder is presented. The remainder of the paper is organized as follows. Section 2 recalls the basic principles of turbo decoding and stochastic computation. Details of multiple stream decoding of turbo codes are given in Section 3. Synthesis and turbo decoding performance results for a turbo decoder are finally presented.

2. STOCHASTIC IMPLEMENTATION OF THE APP ALGORITHM

For convolutional turbo codes, the decoding is performed using the BCJR algorithm, also known as the MAP algorithm [16]. It was adapted by Anderson and Hladik to deal with tail-biting codes. A sub-optimal version of the MAP algorithm in the logarithmic domain with an acceptable loss of performance referred to as Sub-MAP algorithm was then introduced. The stochastic decoding of turbo codes requires the stochastic computation to be applied to a tail-biting A Posteriori Probability (APP) algorithm, which relies on the trellis representation. Fig. 1 details the exchange of information between the various sections of a tail-biting APP decoder.

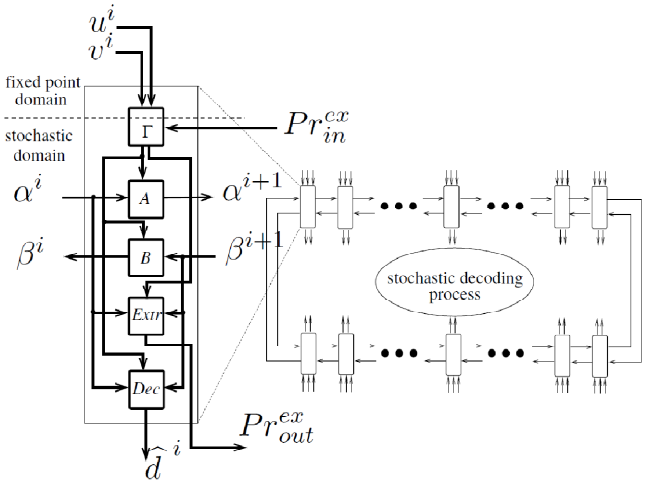

Fig.1. Stochastic tail-biting APP decoder

There are as many sections as symbols to decode and each section is made up of five modules. A Γ module is fed by the channel outputs u^i and v^i, which are associated with the ith transmitted symbol d^i and its parity bit y^i. This module converts u^i and v^i into a priori probabilities, represented by two stochastic streams to compute the branch metrics and then the forward metrics in an *A* module and the backward metrics in a *B* module. These modules are involved in a recursive process since they use the forward and backward metrics α^i and β^{i+1} from their neighbors and provide them α^{i+1} and β^i. A *Dec* module decides the final value of each binary symbol, d^i for the transmitted symbol d^i. A last module is also required if the APP decoder is part of a turbo decoder: the *Extr* module. This module computes the output extrinsic probability $Pr^{ex}{}_{out}$ which is then used by a module Γ of the second APP decoder as the input $Pr^{ex}{}_{in}$. In stochastic decoding, probabilities received from the channel are transformed to streams of stochastic bits using Bernoulli sequences. Each bit in the stream is likely to be '1' with the probability to be transformed. All the modules exchange stochastic streams over a logic gate network based on the code trellis representation. Each stochastic decoding step is referred to as a decoding cycle (DC) and corresponds to the output of one new bit for each stochastic unit. The decoding process terminates when a maximum number of DCs is reached.

Using stochastic representation, operations on probabilities are transformed into bit-serial operations on stochastic streams using simple processing elements. Multiplication and division are thus processed by simple logic gates. From the equations of the APP algorithm, it can be noted that besides the multiplication and division operations, a huge number of additions is necessary. Since the addition of values in the interval [0, 1] may take values bigger than 1, this operation cannot be done directly with stochastic streams. In practice, a multiplexer can produce an output stream that is the scaled sum of the input probabilities by randomly selecting one of the inputs. But, as the output sequence length has to be larger than the input sequence lengths to achieve the same precision, it severely slows down the decoding convergence speed. For this reason, a novel technique for implementing the stochastic addition operation has been proposed in [14]. It consists in transforming the stochastic additions into stochastic multiplications in the exponential domain. The number of DCs is thus considerably reduced with no performance degradation.

3. MULTIPLE STREAM DECODING OF TURBO CODES

Different solutions have been suggested to solve the latching problem, and thus, to improve the BER performance of stochastic decoding, such as: using supernodes, scaling the received Log-Likelihood Ratios (LLRs) up to a maximum value, Edge Memories (EMs) insertion and Noise-Dependent Scaling (NDS). Basically, these solutions aim at re-randomizing and decorrelating stochastic streams. An EM is a complex unit based on a register in which only valuable bits referred to as regenerative bits, i.e. avoiding signals stuck at '0' or '1', are written and randomly read. Such a unit is efficient to solve the latching problem when the register depth is sufficient (typically between 32 and 64) and when it is duplicated for any vertex of the decoding graph, as detailed in [17]. It means that EMs insertion is costly in terms of hardware resources. To improve that, some alternatives have been proposed such as the Tracking Forecast Memory (TFM) [17] and the Majority-based TFM (MTFM) [8]. Nevertheless, these solutions still require some hardware resources such as 8-bit registers, random address

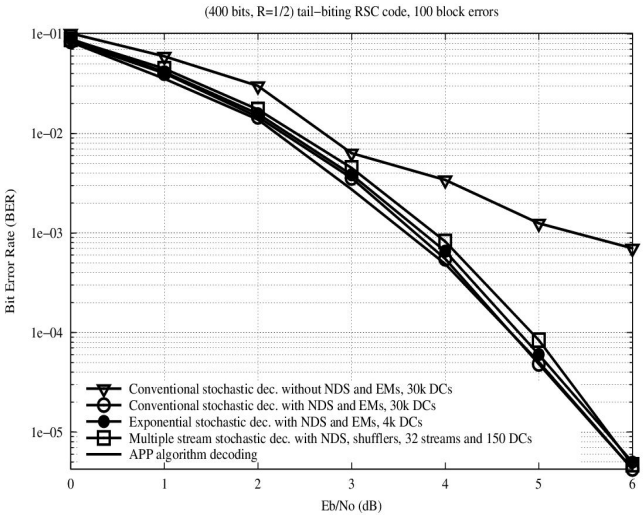

Fig.2. Stochastic decoding performance of convolutional code

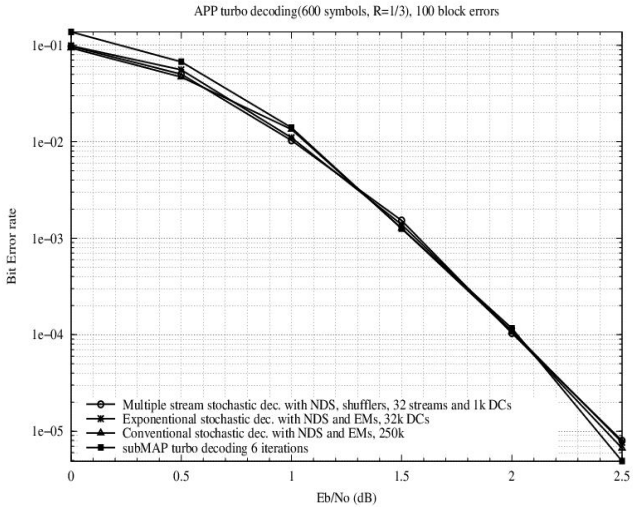

Fig.3. Stochastic decoding performance of turbo code

generators, adders and comparators. Re-randomizing and decorrelating stochastic streams can be done without these complex units if all the probabilities are represented by multiple stochastic streams.

An EM picks up a regenerative bit from a pool when correlation occurs. It is also possible to pick up a regenerative bit from another independent stochastic stream representing the same probability. To reduce the correlation between the concurrent streams, more than two are required. Moreover, the regenerative bit has to be randomly selected among them. In a multiple stream architecture, all the streams and the logic gates are duplicated p times (p>2) and the random bit selection is done by a simple shuffler. The shuffler is made of a register of p JK flip-flops – providing the stochastic normalization –, a p-bit barrel-shifter and p multiplexers. Note that the barrel-shifter is purely combinatorial with a circular shuffling rule to avoid additional random generators. The major interest of multiple stream decoding is that representing a probability by p stochastic streams instead of one divides the number of DCs by p to insure the same precision. It means that a stochastic architecture based on multiple streams has a throughput similar to any other stochastic architecture parallelized at degree p as suggested in [3].

Four digital decoder behavioral models were written in C language to assess the decoding performance of the four decoding methods, namely the proposed multiple stream exponentional stochastic decoder, an exponentional stochastic decoder, a conventional stochastic decoder and a floating-point Sub-MAP decoder for both convolutional and turbo codes. Fig. 2 shows the BER performance of the stochastic decoding of a tail-biting RSC code (n = 400 bits, code rate R = 1/2). A decoder combining NDS and EMs provides a BER performance similar to the one of a conventional APP floating-point algorithm. Processing additions in the exponential domain enables a decrease of the number of DCs from 30K to 4K. Moreover, a 32-stream decoder with shufflers achieves similar BER performance and necessitates only 150 DCs. The stochastic decoding performance of a (n = 600 bits, R = 1/3) turbo code are given in Fig. 3. Conventional stochastic decoder based on EMs and NDS techniques achieves similar BER performance of a floating-point Sub-MAP turbo decoder with 6 iterations. The exponential stochastic approach enables the number of DCs to be reduced from 250K to 32K. The 32-streams stochastic decoder with shufflers requires only 1K DCs. It means that multiple stream architectures for stochastic turbo decoding are competitive in terms of BER performance but also in terms of DCs with state-of-the-art high-speed turbo decoders.

4. FPGA IMPLEMENTATION AND PERFORMANCE RESULTS

Designing stochastic decoding architectures for turbo codes is a challenging issue. In this section, an implementation of a turbo decoder based on the proposed multiple stream exponentional stochastic decoding algorithm is detailed. The additional cost of addition operations in the exponential domain in terms of hardware resources is reasonable as explained in [14]. Indeed, expanding the Taylor series to the second order is sufficient for both exponential and logarithmic modules. Note that a stochastic architecture where all the probabilities are represented by eight independent streams has been designed. It offers a compromise between data throughput and hardware complexity for this first FPGA-based integration. Implementation results of the turbo decoder for a (n=40 bits, R=1/3) turbo code is given in Table I. First, the complexities of one section of the stochastic APP decoder in terms of resources for stochastic computation and for random bit generation are summarized. Remember that a randomization engine is necessary for providing random bits. These random bits are used in 2-to-1 multiplexers and as the addresses of stochastic stream generators. Fortunately, the

complexity impact of the random bit generation can be greatly reduced by reusing them in different modules of the APP decoder and in the two decoders. In our design, 112 slice LUTs and 1,344 slice Flip-Flops are necessary for one section of APP decoder. Then, the hardware complexity of an APP decoder that contains 40 sections is given. Indeed, there are as many sections as symbols to process in the proposed fully-parallel architecture. Finally, the stochastic turbo decoder occupies 149,020 slice LUTs and 90,760 slice Flip-Flops.

Virtex5 LX330		stochastic computation		random bit generation	
		LUT	Flip-Flop	LUT	Flip-Flop
one section of APP decoder	**Γ**	80	0	112	269
	A/B	1,360	768		
	Extr	216	168		
	Dec	152	48		
APP decoder (40 sections)		72,320	39,360	4,480	10,760
turbo decoder (with interleaver)		144,640	80,000		

Table.I. FPGA implementation results of the 8-streams stochastic decoder of a (n=40 bits, R=1/3) turbo code

In order to validate the designed stochastic turbo decoder, BER performance measures have to be carried out. For this reason, we have integrated the designed turbo decoder into an experimental setup. All the components of the experimental setup were implemented onto one Xilinx Virtex5 LX330 FPGA device. A Pseudo Random Generator (PRG) sends out pseudo random data streams at each clock period f_0. This module is composed of flip-flops and XOR gates. A (n=40, R=1/3) turbo code encoder processes the data streams. The last task of the transmitter is a BPSK mapping. We have chosen to integrate an AWGN channel from a white Gaussian noise generator adapted to hardware implementation. The Wallace method generates a variable that is normally distributed from a pool of variables that follows a normal distribution as well [18]. The Signal to Noise Ratio (SNR) is controllable via a computer that sends it to the experimental setup thanks to a PCI bus. Simulated performance of stochastic decoding algorithm and measured performance of designed 8-streams stochastic decoder for the (n=40 bits, R=1/3) turbo code are considered. The prototype shows identical performance when compared to simulation.

CONCLUSION

The paper discussed the architecture of a stochastic turbo decoder. The number of decoding cycles is considerably reduced with no performance degradation by transforming the stochastic additions into stochastic multiplications in the exponential domain and using multiple streams with deterministic shufflers. An FPGA-based architecture for a fully-parallel 8-streams stochastic decoder of a (n = 40, R = 1/3) turbo code is then presented. To the best of our knowledge, this is the first hardware implementation of a stochastic turbo decoder.

REFERENCES

[1] B. Gaines, *Stochastic computing*, in AFIPS SJCC, n° 30, 1967.

[2] W. Poppelbaum, C. Afuso, and J. Esch, *Stochastic computing elements and systems*, in AFIPS FJCC, n°31, 1967.

[3] V. Gaudet and A. Rapley, *Iterative decoding using stochastic computation*, Electronics Letters, vol. 39, n° 3, Feb. 2003.

[4] W. Gross, V. Gaudet, and A. Milner, *Stochastic implementation of LDPC decoders*, in Signals, Systems and Computers, 2005. Conference Record of the Thirty-Ninth Asilomar Conference on, Oct. 28 - Nov. 1 2005, pp. 713–717.

[5] S. Sharifi Tehrani, W. Gross, and S. Mannor, *Stochastic decoding of LDPC codes*, Communications Letters, IEEE, vol. 10, no. 10, pp. 716–718, Oct. 2006.

[6] S. Sharifi Tehrani, S. Mannor, and W. J. Gross, *An area-efficient FPGA-based architecture for fully-parallel stochastic LDPC decoding*, in the Proc. of the IEEE Workshop on Signal Processing Systems (SiPS), Shanghai, China, Oct. 2007.

[7] S. Sharifi Tehrani, S. Mannor, and W. Gross, *Fully parallel stochastic LDPC decoders*, Signal Processing, IEEE Transactions on, vol. 56, no. 11, pp. 5692–5703, Nov. 2008.

[8] S. Tehrani, A. Naderi, G.-A. Kamendje, S. Hemati, S. Mannor, and W. Gross, *Majority-based tracking forecast memories for stochastic LDPC decoding*, Signal Processing, IEEE Transactions on, vol. 58, no. 9, pp. 4883 –4896, Sep. 2010.

[9] S. Sharifi Tehrani, C. Jego, B. Zhu, and W. Gross, *Stochastic decoding of linear block codes with high-density parity-check matrices*, Signal Processing, IEEE Transactions on, vol. 56, no. 11, pp. 5733–5739, Nov. 2008.

[10] V. C. Gaudet and W. J. Gross, *Switching activity in stochastic decoders*, in Proceedings of the 40th IEEE International Symposium on Multiple-Valued Logic (ISMVL), May 2010.

[11] C. Winstead and S. Howard, *A probabilistic LDPC-coded fault compensation technique for reliable nanoscale computing*, Circuits and Systems II: Express Briefs, IEEE Transactions on, vol. 56, no. 6, pp. 484 –488, June 2009.

[12] C. Berrou, A. Glavieux, and P. Thitimajshima, *Near shannon limit error correcting coding and decoding: Turbo-codes*,. ICC 93. IEEE International Conference on, vol. 2, May 1993.

[13] C. Winstead, V. Gaudet, A. Rapley, and C. Schlegel, *Stochastic iterative decoders*, in Proc. IEEE Int. Symp. on Information Theory 2005, pp. 1116–1120.

[14] Q. T. Dong, M. Arzel, C. Jego, and W. J. Gross, *Stochastic Decoding of Turbo Codes*, Signal Processing, IEEE Transactions on, Volume 58, Issue 12, December 2010.

[15] M. Arzel, C. Lahuec, C. Jego, W. J. Gross and Y. Bruned, Stochastic multiple-stream decoding of Cortex codes, submitted at Signal Processing, IEEE Transactions on, vol. PP, no.99, pp.1, 0 doi: 10.1109/TSP.2011.2138699.

[16] L. Bahl, J. Cocke, F. Jelinek, and J. Raviv, "Optimal decoding of linear codes for minimizing symbol error rate," IEEE Trans. Inf. Theory, vol. IT-20, pp. 284-287, Mar. 1974.

[17] S. Tehrani, A. Naderi, G.-A. Kamendje, S. Mannor, and W. Gross, *Tracking forecast memories in stochastic decoders*, in ICASSP 2009, IEEE International Conference on, April 2009.

[18] O. Sanchez Gonzalez, M. Arzel, C. Jégo, A. Garcia, M. Guerrero, "Design and implementation of a MIMO channel emulator onto FPGA device", *IWS'09: XV proyecto Iberchip*, 25-27 March, 2009.